# An Apparatus and Method for Scaling TCP Off Load Buffer Requirements by Segment Size

## FIELD OF THE INVENTION

**[0001]** This invention relates generally to networked communications and, more particularly, relates to a system and method for managing the buffering of data in systems having transport layers that do not have self-describing transport segments.

## BACKGROUND OF THE INVENTION

**[0002]** Computer networking allows applications residing on separate computers or devices to communicate with each other by passing data across the network connecting the computers. The amount of data segments being sent across networks is rapidly increasing. The increase is due to several factors. These factors include the growth in the number of networked computers and the explosion in growth of digitally based multimedia, which is the combination of video and audio in a digital format for viewing on a computer or other digital device.

**[0003]** Data is typically sent in segments or packets. One of the problems that has occurred as a result of the increased data flow is segment delay, segment loss and reordering of segments (e.g., segments arriving out of order). The segment delay, segment loss, and reordering of segments causes problems for many applications that rely on the data being received timely and in the order it was sent. For example, applications such as digitally based multimedia that receive data sent in segments must receive the data in the order the segments were sent and without any delay for proper playback.

**[0004]** Many data transmission protocols account for segment delay, loss, and reordering by placing the data in an intermediate reassembly buffer to restore the sending order which may have been lost as a result of segment delay, loss, or reordering. If data comes in order, the data is put directly in a destination buffer. If data does not come in order, the data is put into an intermediate reassembly buffer. The bandwidth required to move data in and out of the buffer memory of the intermediate reassembly buffer is more than twice the bandwidth of the incoming data. The size of the buffer memory required is:

K*(window_size)*(# connections)

where K is a constant. Window_size is generally assumed to be the bandwidth delay product, which is the round trip time (RTT) of data sent between the sending computer and the receiving computer multiplied by the bandwidth of the data. As the speed of networks increases, the required speed and size of the buffer also increases. For example, on a local area network running at one gigabit/sec with a one millisecond RTT having 100 simultaneous connections, the memory required is 12.5 MB. For wide area networks running at ten gigabits/sec and having 100 millisecond RTT and 100 simultaneous connections, the total buffer memory required is 12.5 GB.

**[0005]** The required buffer memory is a problem when the Transmission Control Protocol (TCP) is offloaded from the host to a network interface card (NIC). When the TCP is running on a NIC, the intermediate reassembly buffer is typically placed on the NIC connection (i.e., on the network interface card). Today, network interface cards having megabytes or gigabytes of memory are not cost competitive due to the memory size and high speed requirements.

**[0006]** One approach that the computer industry has taken to reduce the amount of memory needed is the development of upper layer protocols (ULPs), such as those which perform bulk data transfer, that permit the final location (i.e., a client's buffer) to be known when the data is received. An upper layer protocol is a protocol that resides above the data transmission protocol layer in the computer or other digital device. The ULPs that perform bulk data transfer include specific application ULPs such as iSCSI and generic hardware acceleration ULPs such as a remote direct memory access (RDMA) protocol. When an upper layer protocol that permits the final location to be known is used, the data can be directly placed at the final location by a direct placement network interface that has knowledge of the final location.



**[0007]** The information required to find the final location is contained in the upper layer protocol protocol data unit (ULP PDU) header. A ULP PDU is an upper layer protocol data packet or message. However, the ULP PDU is not always identical to the size of the transport segment used by the transport protocol to send the data over the network. When this occurs, the ULP PDU is split and placed into several transport segments. This results in the ULP PDU header not being located in every transport segment and located at any place in a transport segment. The transport segments that do not contain the ULP PDU header are called non self-describing transport segments because they do not have sufficient information to directly place the data into its final location. The ULP PDU header must be located for direct placement to work.

**[0008]** One solution to finding the ULP PDU header in the transport segment (i.e., transport data packet) is to put a marker in each transport segment at a predefined period that provides a pointer to ULP PDU header of the next PDU. If the period is high relative

to packet loss rate (e.g., once per segment), the location of the next PDU can be found with a very high likelihood when a previous PDU header has been lost. The problem with this solution is that data must still be buffered. The minimum size of the buffer required is:

k*(the size of the ULP PDU) * (the number of connections)

where k is a constant. If the size of the ULP PDU is limited to the transport segment size, the minimum buffer size required is:

(maximum transport segment size) * (the number of connections)

In cases where many transport segments having ULP PDU headers are lost, the buffer size required for direct placement approaches the intermediate reassembly buffer sizes of multiple gigabytes as indicated above.



## BRIEF SUMMARY OF THE INVENTION

**[0009]** The present invention provides a method to ensure that there is enough information in each transport segment to allow the data being sent in the transport segment to be placed in the client's buffer. A framing header is aligned with the transport segment header so that the framing header is at the beginning of the transport segment. Information is placed in the framing header that describes the transport segment in sufficient detail to allow the data in the transport segment to be placed in its final destination. This effectively eliminates the need for an intermediate reassembly buffer in the performance path (i.e., the primary path in which the data is being received).

**[0010]** The information includes the length of the transport segment (i.e., the size), an offset and the ULP client buffer id. In an alternate embodiment, the information includes the ULP client buffer id and the ULP client buffer address.

**[0011]** Additional features and advantages of the invention will be made apparent from the following detailed description of illustrative embodiments which proceeds with reference to the accompanying figures.



## BRIEF DESCRIPTION OF THE DRAWINGS

**[0012]** While the appended claims set forth the features of the present invention with particularity, the invention, together with its objects and advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

**[0013]** Figure 1 is a block diagram illustrating a network system having a plurality of networks connected by a transport in which the present invention operates;

**[0014]** Figure 2 is a block diagram generally illustrating an exemplary computer system on which the present invention resides;

**[0015]** Figure 3 is a block diagram illustrating an embodiment of a network protocol stack employing a framing protocol in accordance with the teaching of the present invention;

**[0016]** Figure 4 is a block diagram illustrating an alternate embodiment of an operating system employing a framing protocol in accordance with the teaching of the present invention;

**[0017]** Figure 5 is a diagram illustrating an example of a framing header aligned with a transport segment header in accordance with the present invention;

**[0018]** Figures 6a and 6b illustrate a flow chart of a method of transforming a non self-describing protocol data unit into a self-describing protocol data unit in accordance with the present invention; and

**[0019]** Figure 7 is a diagram illustrating the format of a framing data unit in accordance with the present invention.



## DETAILED DESCRIPTION OF THE INVENTION

**[0020]** The present invention ensures that ULP PDUs received out of order do not have to be buffered by an intermediate reassembly buffer in the performance path. To provide this ensurance, each transport segment begins with segment description information and contains an integral number of ULP PDUs. Segment description information is information that is sufficient for direct data placement to occur. Direct data placement occurs when a receiving node directly maps the incoming data (e.g., the data in the ULP PDU) to a specific ULP buffer and a specific offset within that buffer. In this way, the destination location of the ULP PDU is known and the out of order ULP PDU is placed in its destination location. The size of the buffer memory required is reduced to:

k* (the maximum transport segment size)

where k is a constant. In systems having direct placement capability, the present invention allows the data in each transport segment to be placed in its final destination

independently of all other segments and requires no intermediate reassembly buffers in the performance path.

**[0021]** Although not required, the invention is described hereinafter in the general context of computer-executable instructions, such as program modules, being executed by a processing device. Generally, program modules include routines, programs, objects, components, data structures, and the like that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including network interface cards, hand-held devices, multi-processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

**[0022]** Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable networking environment such as the network system environment 20 illustrated in Figure 1. The network system environment 20 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. For example, those of skill in the art will appreciate that the invention can also be used in wireless systems and that various of the acts and operations described hereinafter may also be implemented in hardware.

**[0023]** The exemplary network system 20 includes local sub-networks 22, 24 connected to each other by network 26. Network 26 may be a virtual private network, a LAN, a WAN, the Internet, and the like. Local sub-network 22 has general purpose computing devices in the form of computers 28, 30, and local sub-network 24 has computer 32. The network system 20 may have additional local sub-networks and local sub-networks 22, 24 may have additional computers. Each sub-network 22, 24 is served by a transport provider 34, such as TCP/IP, for providing communication between computers and between applications residing on computers.



**[0024]** An example of a suitable computing system environment 100 on which the invention may be implemented is illustrated in Figure 2. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

**[0025]** The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, network cards, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

**[0026]** The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a processing device. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

**[0027]** With reference to Figure 2, an exemplary system for implementing the invention includes a general purpose computing device in the form of a processing device 110. It should be noted that computers 28, 30, 32 are similar to processing device 110 and have similar components. In its most basic configuration, processing device 110 typically includes a processing unit 112 and system memory 114. Processing unit 112 spans a diverse set of implementations including a central processing unit, a hardware finite state machine based implementation, and a microprocessor assembly language implementation. Depending on the exact configuration and type of computing device, memory 114 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. Memory 114 has buffers 116 for receiving, sending, and storing data. Additionally, processing device 110 may also have removable mass storage 118 and/or non-removable mass storage 120 such as magnetic disks, magnetic cassettes, optical disks or tape. Similarly, processing device 110 may also have input devices 122 such as a mouse, a keyboard, a modem, and the like and /or output devices 124 such as a display, a printer, speakers and the like. Other aspects of

processing device 110 may include communication device 126 for communicating to other devices, computers, networks, servers, etc. using either wired or wireless media or a combination of the above. Network communication occurs through communication device 126 and may include buffers 128 on the communication device 126, buffers 116 in the processing device 110, or a combination of buffers 116 and buffers 128. All of these devices are well known in the art and need not be discussed at length here.

**[0028]** Processing device 110 typically includes at least some form of computer readable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory, or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information. Communication media typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and

wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

**[0029]** Figures 3 and 4 illustrate where the framing protocol of the present invention resides on standard network stack (or layer) architectures that have transport layers that use transport protocols. The framing protocol and transport protocol may reside in the processing device 110 or in the communication device 126 or in a combination of the processing device 110 and communication device 126. Figure 3 shows an architectural overview of an embodiment of the framing protocol according to the invention implemented in the International Standard Organization's Open System Interconnect (ISO/OSI) model. In this embodiment, the framing protocol 202 is in communication with upper layer protocol client 200 and with transport layer 204. The framing protocol 202 may be integrated with the transport layer 204. The transport layer 204 resides above the network layer 206 and enforces desired transmission control using a transport protocol. A transport protocol controls the transmission of transport segments. The network layer 206 ensures that information arrives at its intended destination across a diverse set of data link layers. The data link layer 208 resides below the network layer 206 and packages and addresses data and manages the flow of transmissions on a local link. The physical layer 210 resides below the data link layer 208 and deals with all aspects of establishing and maintaining a physical link between communicating computers.

**[0030]** Figure 4 shows an architectural overview of an embodiment of the framing protocol implemented in the TCP/IP environment. The framing protocol 202 sits above the standard TCP protocol 220, which is a transport protocol, and the IP protocol 222.

The IP protocol 222 is in communication with the network interface card (NIC) 226 via NDIS 224. While the framing protocol 202 has been illustrated as a separate layer in figures 3 and 4, the framing protocol 202 may be integrated with other layers.

**[0031]** In the description that follows, the invention will be described with reference to acts and symbolic representations of operations that are performed by one or more computers, unless indicated otherwise. As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computer of electrical signals representing data in a structured form. This manipulation transforms the data or maintains it at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art. The data structures where data is maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that various of the acts and operation described hereinafter may also be implemented in hardware.

**[0032]** Now that the architecture in which the invention operates and resides has been described, the steps taken to ensure that each transport segment begins with an ULP PDU (i.e., the ULP PDU is aligned with the transport segment) and contains an integral number of ULP PDUs can be described as illustrated in Figure 5. Turning now to Figure 5, each transport segment 300 has a transport header 302. The invention places a framing header 304 having segment description information within or after the transport header 302. The segment description information describes the data 306 with sufficient detail so

that data in transport segments received out of order can be placed in its destination location . Transport segments that have the segment description information are called self-describing segments.

**[0033]** Turning now to the flow diagram of Figures 6a and 6b, the framing protocol receives and processes the ULP PDU (step 400) from an upper layer sender (e.g., an application, a device, a ULP, etc.) to hand off to the transport layer to send to an upper layer receiver. The framing protocol uses buffers 116, buffers 128, or a combination of buffers 116, 128 to receive and process the ULP PDU. The framing protocol first determines if the size of the ULP PDU fits within predetermined limits (step 402). The limits are selected based upon the transport segment size being used, the maximum transport segment size defined by the protocol used, and the degree that the transport protocol supports chunking. A chunking operation is an operation that splits the ULP PDU into packets that fit within the current transport segment size. The limits are provided to the framing protocol by the transport protocol. The framing protocol is notified when the transport segment size presently being used changes.

**[0034]** If the ULP PDU size submitted by the ULP is outside the limit, the framing protocol fails the submission and generates an error message (step 404). In an alternate embodiment, the framing protocol performs a chunking operation on the ULP PDU when the ULP PDU size is larger than the limit. The chunking operation splits the ULP PDU into packets that fit within each transport segment and splits the ULP PDU into multiple transport segments. The offset within the buffer from the base of the buffer and/or the buffer address is incremented in the transport segments so that direct data placement can occur.

**[0035]** The length (i.e., size) of the ULP PDUs packed in the transport segment is determined by the framing protocol (step 406). If the ULP PDU size is within the limit, the framing protocol determines if multiple ULP PDUs fit within a transport segment (step 408). If multiple ULP PDUs will fit within the transport segment, the framing protocol determines if a single PDU option was chosen (step 410). The single PDU option is selected by the receiver (e.g., receiving application, protocol, etc.) or the transmitter (e.g., sending application, protocol, etc.). When this option is selected, only one ULP PDU may be packed into a transport segment.

**[0036]** If multiple ULP PDUs do not fit within a transport segment or the single PDU option was chosen, the framing protocol puts one ULP PDU into a transport segment (step 412). If multiple ULP PDUs fit within a transport segment and the single PDU option was not chosen, an integral number of ULP PDUs are put into a transport segment (step 414).

**[0037]** Information that self-describes the transport segment is placed into the transport header or right after the transport header in a framing header (step 416) and the transport segment is sent to the receiver (step 418). This information, which is called segment description information, includes the length and an identifier that identifies the connection. This information provides a receiver with sufficient information to determine where the ULP PDU should be placed. This allows successive ULP PDUs to be recovered in the event of transport segment delay, loss, or reordering. In the event direct placement networks are being used, further information is put in the transport segment to allow the transport payload (i.e., the ULP PDU(s)) to be put in its final destination. The

additional information includes the buffer id where the data is to be placed and the offset (i.e., the offset within the buffer from the base of the buffer).

**[0038]** Turning now to figure 7, the format of a self-describing transport segment 500 is illustrated. The length field 502 and identifier field 504 are part of the framing header 304. Following the framing header is at least one integral ULP PDU in the payload 306. The direct placement information (buffer id and offset) is also placed in the framing header 304.

**[0039]** There are exceptions to sending an integral ULP PDU in a transport segment. These exceptions include situations when the connection is in emergency mode, when the transport maximum segment size has been reduced, and when the sender is probing to determine if the maximum transport segment size unit can be increased. A connection is in emergency mode when the transport segment size being used shrinks to a size which leaves no room for a complete ULP PDU to be sent. When an emergency mode occurs, the framing protocol may fragment an ULP PDU across multiple transport segments. The transport segments having a fragment of an ULP PDU should not appear to be a complete ULP PDU to the receiving application/device. There are many ways to ensure this occurs. For example, in one embodiment, this is ensured by the sender making sure that either the data in the length field does not equal the transfer segment length, or that the transfer segment length is less than the framing header in each transport segment.

**[0040]** As previously indicated, the present invention allows the data in each transport segment to be placed in its final destination independently of all other segments and requires no intermediate reassembly buffers in the performance path. A reassembly buffer is needed if synchronization of the framing header with the transport header is lost

or if there is a program module or hardware device (i.e., a middle box) re-segmenting the data stream in the network. The former example (i.e., loss of synchronization) is a transient condition that only occurs when a path transport segment size being used has been reduced and there is no middle-box in the network which is re-segmenting the data stream. If the path transport segment size being used is reduced, any transport segments already converted to self-describing segments and posted to the transmitter will be re-segmented and will result in non self-describing segments being sent to the destination. If the destination does not use a reassembly buffer, these non self-describing segments can never be successfully transmitted. The reassembly buffer must be large enough to recover a transport segment that is less than or equal to the maximum transport segment size of the locally attached link.

**[0041]** The invention has been described with some references to the TCP protocol. The invention may also be used with other protocols, such as SCTP, that use transport segments that do not describe in sufficient detail where the payload is to be placed.

**[0042]** In view of the many possible embodiments to which the principles of this invention may be applied, it should be recognized that the embodiment described herein with respect to the drawing figures is meant to be illustrative only and should not be taken as limiting the scope of invention. For example, those of skill in the art will recognize that the elements of the illustrated embodiment shown in software may be implemented in hardware and vice versa or that the illustrated embodiment can be modified in arrangement and detail without departing from the spirit of the invention. Therefore, the invention as described herein contemplates all such embodiments as may come within the scope of the following claims and equivalents thereof.